案例 17-直线 Cohen-Sutherland 裁剪算法

文档编写：霍波魏

校稿/修订：孔令德

时间 2019~2020

联系方式：QQ997796978

说明：本套案例由孔令德开发，原版本为 Visual C++6.0，配套于孔令德的著作《计算机图形学-基于 MFC 三维图形开发》一书。孔令德计算机工程研究所的学生霍波魏在学习计算机图形学期间，对本套案例进行了升级并编写了学习文档。现在程序的编写和程序的解释都是基于 Windows 10 操作系统，使用 Microsoft visual studio 2017 平台的 MFC（英文版）开发。

一、知识点

1. 编码原理

每段直线的端点都被赋予一组 4 位二进制代码，称为区域编码，用来标识直线端点相对于窗口边界及其延长线的位置。

2. 交点计算公式

设端点坐标为$\mathrm{P_0(x_0,y_0)}$, $\mathrm{P_1(x_1,y_1)}$,与窗口左边界$(\mathrm{x}=\mathrm{w}_{xl})$或右边界$(\mathrm{x}=\mathrm{w}_{xr})$的交点的 y 坐标的计算公式为：

$$y = k(x - x_0) + y_0, \qquad k = (y_1 - y_0)/(x_1 - x_0)$$

与窗口上边界$(\mathrm{y}=\mathrm{w}_{yt})$或下边界$(\mathrm{y}=\mathrm{w}_{yb})$交点的 x 的坐标的计算公式为：

$$x = \frac{y - y_0}{k} + x_0\,, \qquad k = (y_1 - y_0)/(x_1 - x_0)$$

二、案例描述

本案例使用 Cohen-Sutherland 裁剪算法，在一个窗口内裁剪所绘制的直线。

三、实现步骤

1. 添加绘制直线类 Cline 类
2. 在 CTestView 中添加编码函数、绘制裁剪窗口函数、坐标变换以及裁剪函数
3. 在 CTestView 中添加消息函数，在 OnDraw 中调用 DoubleBuffer 函数定义客户区坐标、绘制裁剪窗口

4. 在 OnClip()函数中调用裁剪函数，进行直线绘制的判断以及直线的裁剪

四、主要算法

CTestView 类

```
public:
    void DoubleBuffer(CDC* pDC);//双缓冲函数
    void DrawWindowRect(CDC* pDC);//绘制裁剪窗口函数
    CP2i Convert(CPoint point);//坐标系变换
    void CSLineClip();//Cohen-Sutherland直线段裁剪函数
    void EnCode(CP2i &pt);//编码函数
protected:
    int nClientWidth, nClientHeight;//屏幕客户区宽度和高度
    int nHWidth, nHHeight;//屏幕客户区的半宽和半高
    CLine* line;//直线的指针
    CP2i P[2];//直线的起点和终点
    int  PtCount;//顶点个数
    int  Wxl, Wxr;//窗口左上角点
    int  Wyb, Wyt;//窗口右下角点
    BOOL bDrawLine;//是否允许画线
#define LEFT      0x0001   //代表0001
#define RIGHT     0x0002   //代表0010
#define BOTTOM    0x0004   //代表0100
#define TOP       0x0008   //代表1000
void CTestView::DoubleBuffer(CDC* pDC)//双缓冲
{
    CRect rect;//定义客户区
    GetClientRect(&rect);//获得客户区的大小
    nClientWidth = rect.Width();//屏幕客户区宽度
    nClientHeight = rect.Height();//屏幕客户区高度
    nHWidth = nClientWidth / 2;//屏幕客户区半宽
    nHHeight = nClientHeight / 2;//屏幕客户区半高
    CDC memDC;
    memDC.CreateCompatibleDC(pDC);
    CBitmap NewBitmap, *pOldBitmap;
    NewBitmap.CreateCompatibleBitmap(pDC, nClientWidth, nClientHeight);
    pOldBitmap = memDC.SelectObject(&NewBitmap);
    memDC.FillSolidRect(&rect, pDC->GetBkColor());
                        //按原来背景填充客户区，否则是黑色
    DrawWindowRect(&memDC);//绘制裁剪窗口
    if (PtCount >= 1)
    {
        line->MoveTo(&memDC, nHWidth + P[0].x, nHHeight - P[0].y);
        line->LineTo(&memDC, nHWidth + P[1].x, nHHeight - P[1].y);
```

```
    }
    pDC->BitBlt(0, 0, nClientWidth, nClientHeight, &memDC, 0, 0, SRCCOPY);
                    //将内存位图拷贝到屏幕
    memDC.SelectObject(pOldBitmap);//恢复位图
    NewBitmap.DeleteObject();//删除位图
}
void CTestView::DrawWindowRect(CDC* pDC)//绘制裁剪窗口函数
{
    pDC->SetTextColor(RGB(128, 0, 0));
    pDC->TextOut(nHWidth - 10, nHHeight - Wyt - 20, _T("窗口"));
    CRGB LineClr = CRGB(0.0, 0.5, 0.0);
    line->MoveTo(pDC, nHWidth + Wxl, nHHeight - Wyt, LineClr);
    line->LineTo(pDC, nHWidth + Wxr, nHHeight - Wyt, LineClr, 3);
    line->LineTo(pDC, nHWidth + Wxr, nHHeight - Wyb, LineClr, 3);
    line->LineTo(pDC, nHWidth + Wxl, nHHeight - Wyb, LineClr, 3);
    line->LineTo(pDC, nHWidth + Wxl, nHHeight - Wyt, LineClr, 3);
}
CP2i CTestView::Convert(CPoint point)//坐标系变换
{
    CP2i ptemp;
    ptemp.x = point.x - nHWidth;
    ptemp.y = nHHeight - point.y;
    return ptemp;
}
void CTestView::CSLineClip()//Cohen-Sutherland直线段裁剪函数
{
    CP2i p;//交点坐标
    EnCode(P[0]);//起点编码
    EnCode(P[1]);//终点编码
    while (P[0].rc != 0 || P[1].rc != 0)//处理至少一个顶点在窗口之外的情况
    {
        if ((P[0].rc & P[1].rc) != 0)//简弃之
        {
            PtCount = 0;
            return;
        }
        if (0 == P[0].rc)//确保P[0]位于窗口之外
        {
            CP2i pTemp;
            pTemp = P[0];
            P[0] = P[1];
            P[1] = pTemp;
        }
        UINT RC = P[0].rc;
```

```
            double k = double(P[1].y - P[0].y) / (P[1].x - P[0].x);//直线段的斜率
            //窗口边界按左、右、下、上的顺序裁剪直线段
            if (RC & LEFT)//P[0]点位于窗口的左侧
            {
                p.x = Wxl;//计算交点y坐标
                p.y = ROUND(k * (p.x - P[0].x) + P[0].y);
            }
            else if (RC & RIGHT)//P[0]点位于窗口的右侧
            {
                p.x = Wxr;//计算交点y坐标
                p.y = ROUND(k * (p.x - P[0].x) + P[0].y);
            }
            else if (RC & BOTTOM)//P[0]点位于窗口的下侧
            {
                p.y = Wyb;//计算交点x坐标
                p.x = ROUND((p.y - P[0].y) / k + P[0].x);
            }
            else if (RC & TOP)//P[0]点位于窗口的上侧
            {
                p.y = Wyt;//计算交点x坐标
                p.x = ROUND((p.y - P[0].y) / k + P[0].x);
            }
            EnCode(p);
            P[0] = p;
        }
    }
    void CTestView::EnCode(CP2i &pt)//编码函数
    {
        pt.rc = 0;
        if (pt.x < Wxl)
            pt.rc = pt.rc | LEFT;
        else if (pt.x > Wxr)
            pt.rc = pt.rc | RIGHT;
        if (pt.y < Wyb)
            pt.rc = pt.rc | BOTTOM;
        else if (pt.y > Wyt)
            pt.rc = pt.rc | TOP;
    }
```

五、实现效果

直线 Cohen-Sutherland 裁剪算法效果如图 17-1、17-2 和 17-3 所示。

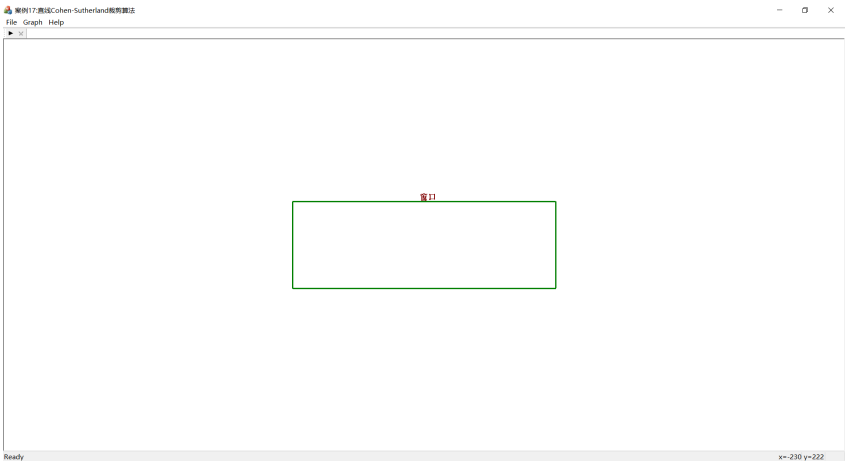

图 17-1 直线 Cohen-Sutherland 裁剪算法效果图一

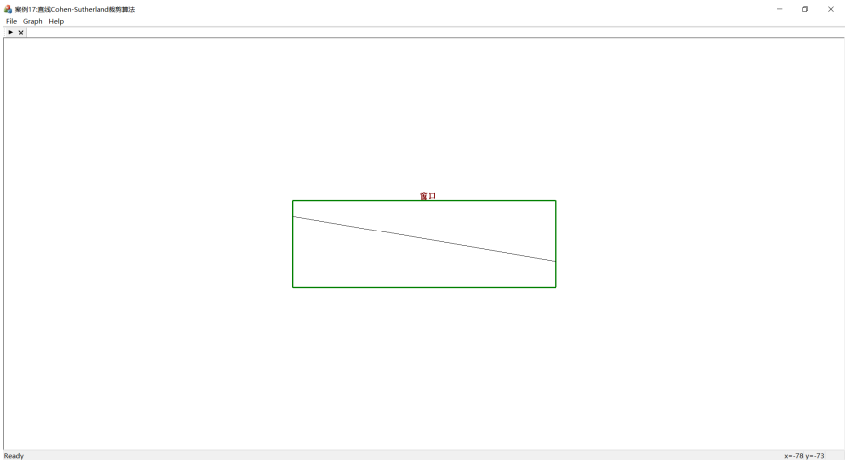

图 17-2 直线 Cohen-Sutherland 裁剪算法效果图二

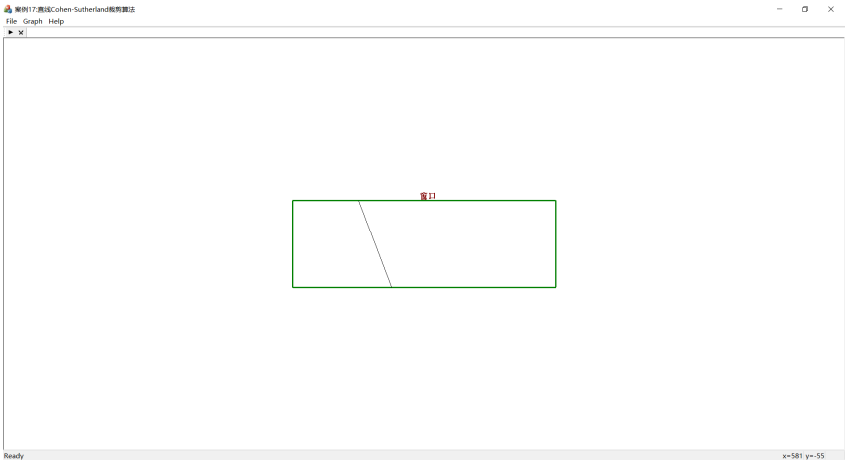

图 17-3 直线 Cohen-Sutherland 裁剪算法效果图三